Jenkins and **selenium Grid** must be configurated.

Everything is happened in the **Cloud AWS**

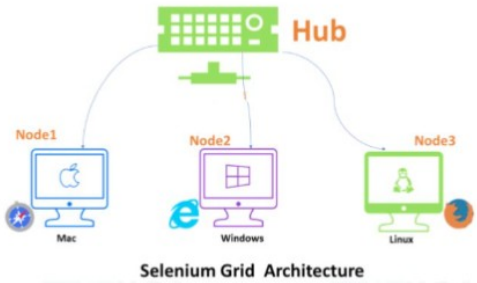
WHY SELENIUM GRID?

- **Capabilities of Selenium Grid:**
  - Parallel Execution: Run tests simultaneously across different machines.
  - Cross-Browser Testing: Ensure compatibility across multiple browsers.
- **Benefits of using Selenium Grid:**
  - Efficient Resource Utilization: Maximizes testing with minimal hardware.
  - Faster Test Execution: Parallel testing means shorter test cycles.
  - Improved Test Coverage: Test on multiple environments concurrently.

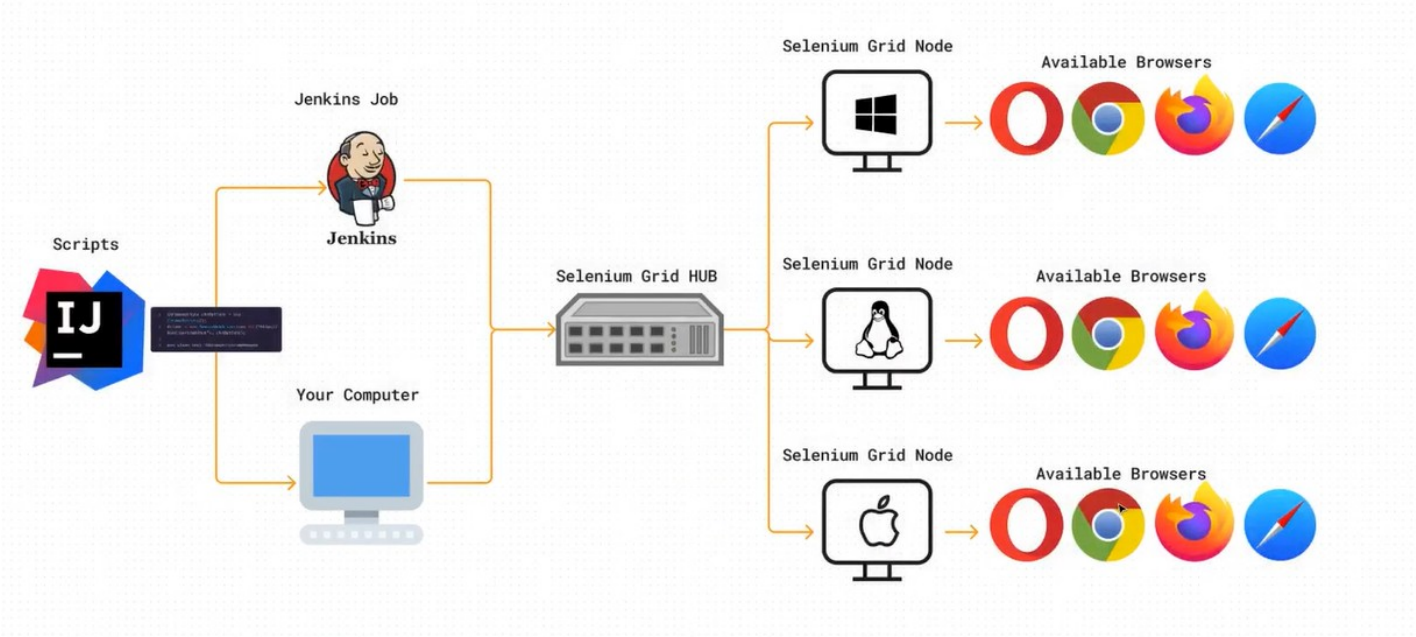
SELENIUM GRID

- Selenium Grid → is used to run multiple tests in **parallel** across **different browsers, operating systems, and machines**.
- It consists of one **Hub VM + multiple Node VMs**

- Hub orchestrates and distributes tests to be run on Nodes. So, tests actually are executed on the Nodes, not on the Hub.
- Nodes can be Windows, Linux, Mac
- Nodes can have different browsers like: Chrome, Firefox, Edge, Safari etc.



Selenium Grid Architecture

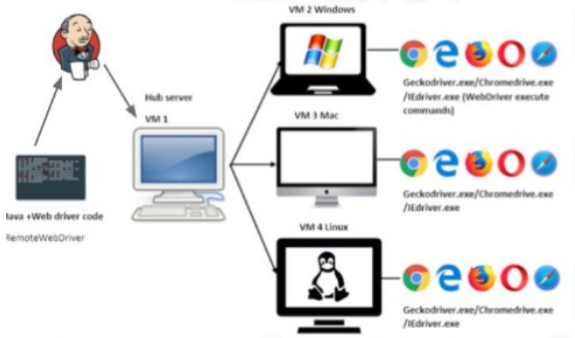



```
1. Parallel Testing:
- Our framework can be run in parallel with surefire plugin.
- Or with more sophisticated tool - Selenium Grid.
- If no above tools our tests will be simply run sequentially.

2. Executing the tests remotely:
- RemoteWebDriver can handle that by privileged the IP address of the machine we want to execute our scripts
- At this point it doesn't really matter if we have any of parallel setup, we can simply do remote execution with sequential run

3. Integration of RemoteWebDriver and Selenium GRID is a powerful combination
- Because we can run the tests on a remote machine and Selenium Grid can orchestrate the execution
```

```
<plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.0</version>
                <configuration>
                    <parallel>methods</parallel>
                    <threadCount>2</threadCount>
                    <testFailureIgnore>true</testFailureIgnore>
                </configuration>
</plugin>
```

Grid Configuration

- We configure our Driver Class with options called RemoteDriver
  - We pass IP address of the Selenium Grid Hub as desired capability
  - `ChromeOptions chrOptions = new ChromeOptions();`
  - `driver = new RemoteWebDriver(new URL("http://34.235.0.4:4444/wd/hub"), chrOptions);}`
- We can trigger build from **Jenkins** by selecting Remote ChromeDriver as parameter
- This way Jenkins will start the execution using Selenium Grid on that provided IP address.
- Tests will get executed on the nodes.




```
package ui_automation.utilities;

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebDriverException;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import org.openqa.selenium.edge.EdgeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.firefox.FirefoxOptions;
import org.openqa.selenium.ie.InternetExplorerDriver;
import org.openqa.selenium.remote.RemoteWebDriver;
import org.openqa.selenium.safari.SafariDriver;

import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.HashMap;

public class BrowserFactory {
    /**
     * the createInstance method will create different browser instances
     * based on the need.
     * If we execute through the command line like below, by specifying the type of
browser
     * `mvn clean test -Dtest=UITestRunner -Dbrowser=chromeRemote`
     * if we don't specify the type of browser from command line, or execute though IDE,
     * JVM won't have browser property
     * <p>
     * mvn clean test -Dtest=UITestRunner
     *
     * @return WebDriver instance
     */
    public static WebDriver createInstance() {
```

```
        WebDriver driver = null;

        try {
            if (System.getProperty("browser") == null) {
                // because of the selenium version 3, we need to specify where is the
exec file
                // Boni Garcia Library is profiding below line to find out during
runtime
                WebDriverManager.chromedriver().setup();
                // Create a key/value of ther preferences
                HashMap<String, Object> chromePrefs = new HashMap<String, Object>();
                String path = System.getProperty("user.dir") +
                        File.separator +
                        "src" +
                        File.separator +
                        "test" +
                        File.separator +
                        "resources" +
                        File.separator +
                        "testData" +
                        File.separator +
                        "Downloads";
                chromePrefs.put("download.default_directory", path);
                // create the Chrome Options
                ChromeOptions options = new ChromeOptions();
                options.addArguments("--ignore-ssl-errors=yes");
                options.addArguments("--ignore-certificate-errors");
                options.setExperimentalOption("prefs", chromePrefs);
                driver = new ChromeDriver(options);
            } else {
                switch (System.getProperty("browser")) {
                    case "chrome-headless":
//Headless Chrome refers to running the Google Chrome browser in a mode that operates
without a graphical user interface (GUI).
                        WebDriverManager.chromedriver().setup();
                        driver = new ChromeDriver(new
ChromeOptions().setHeadless(true));
                        break;
                    case "chromeRemote":
                        HashMap<String, Object> chromePrefs = new HashMap<String,
Object>();
                        chromePrefs.put("download.default_directory",
System.getProperty("user.dir") + "\\src\\test\\resources\\testData\\Downloads");
                        ChromeOptions chrOptions = new ChromeOptions();
                        chrOptions.addArguments("--ignore-ssl-errors=yes");
                        chrOptions.addArguments("--ignore-certificate-errors");
                        chrOptions.setExperimentalOption("prefs", chromePrefs);
                        try {
                            driver = new RemoteWebDriver(new
URL("http://54.80.58.108:4444/wd/hub"), chrOptions);
//It is the computer address when the test is run
//The host and the url is provided by the DevOps
                        } catch (MalformedURLException e) {
                            e.printStackTrace();
                        }
```

```
                    break;
                case "firefox":
                    WebDriverManager.firefoxdriver().setup();
                    driver = new FirefoxDriver();
                    break;
                case "firefox-headless":
                    WebDriverManager.firefoxdriver().setup();
                    driver = new FirefoxDriver(new
FirefoxOptions().setHeadless(true));
                    break;
                case "firefoxRemote":
                    FirefoxOptions firOptions = new FirefoxOptions();
                    try {
                        driver = new RemoteWebDriver(new
URL("http://54.80.58.108:4444/wd/hub"), firOptions);
//It is the computer address when the test is run
//The host and the url is provided by the DevOps

                    } catch (MalformedURLException e) {
                        e.printStackTrace();
                    }
                    break;
                case "ie":
                    if (System.getProperty("os.name").toLowerCase().contains("mac"))
{
                        throw new WebDriverException("Your operating system does not
support the requested browser");
                    }
                    WebDriverManager.iedriver().setup();
                    driver = new InternetExplorerDriver();
                    break;

                case "edge":
                    if (System.getProperty("os.name").toLowerCase().contains("mac"))
{
                        throw new WebDriverException("Your operating system does not
support the requested browser");
                    }
                    WebDriverManager.edgedriver().setup();
                    driver = new EdgeDriver();
                    break;

                case "safari":
                    if
(System.getProperty("os.name").toLowerCase().contains("windows")) {
                        throw new WebDriverException("Your operating system does not
support the requested browser");
                    }
                    WebDriverManager.getInstance(SafariDriver.class).setup();
                    driver = new SafariDriver();
                    break;
                default:
                    WebDriverManager.chromedriver().setup();
                    driver = new ChromeDriver();
```

```
                    break;
                }
            }

        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        return driver;
    }

    public static WebDriver createInstance(String browser) {
        WebDriver driver = null;
        if (browser.equals("chromeRemote")) {
            HashMap<String, Object> chromePrefs = new HashMap<String, Object>();
            chromePrefs.put("download.default_directory", System.getProperty("user.dir")
+ "\\src\\test\\resources\\testData\\Downloads");
            ChromeOptions chrOptions = new ChromeOptions();
            chrOptions.addArguments("--ignore-ssl-errors=yes");
            chrOptions.addArguments("--ignore-certificate-errors");
            chrOptions.setExperimentalOption("prefs", chromePrefs);

            try {
                driver = new RemoteWebDriver(new URL("http://54.80.58.108:4444/wd/hub"),
chrOptions);
            } catch (MalformedURLException e) {
                e.printStackTrace();
            }
        } else {
            driver = new ChromeDriver();
        }
        return driver;
    }
}
```

Look to the link: it is done by the DevOps

Getting started with Selenium Grid | Selenium

Jenkins:

Link:

Sign in [Jenkins]

Selenium Grid is a powerful tool that enables the execution of automated tests across multiple browsers, operating systems, and machines simultaneously. This distributed testing capability is achieved through a hub-node architecture, where the hub acts as a central point controlling various nodes that execute the tests. The `RemoteWebDriver` class plays a crucial role in this setup by facilitating communication between the test scripts and the remote nodes.

RemoteWebDriver:

`RemoteWebDriver` is a class in the Selenium framework that allows test scripts to execute commands on remote browser instances. Instead of running tests locally, `RemoteWebDriver` communicates with remote nodes over a network, sending commands and receiving responses. This setup is essential for distributed testing environments like Selenium Grid.

Setting Up Selenium Grid with a Hub and Multiple Nodes on Virtual Machines:

1. **Prepare the Environment:**
   - **Virtual Machines (VMs):** Set up multiple VMs to act as nodes. Each VM can run different operating systems and browser configurations as needed.
   - **Java Installation:** Ensure that Java is installed on all machines (hub and nodes) since Selenium Server requires Java to run.
2. **Download Selenium Server:**
   - Obtain the latest Selenium Server jar file from the official Selenium website.
3. **Configure the Hub:**
   - **Select a Machine:** Choose one machine to function as the hub.
   - **Start the Hub:** Open a command prompt or terminal on the hub machine and navigate to the directory containing the Selenium Server jar file. Execute the following command:

     ```
     bash
     CopyEdit
     java -jar selenium-server-standalone-<version>.jar -role hub
     ```

     By default, the hub starts on port 4444. You can specify a different port using the `-port` parameter if necessary.

4. **Configure the Nodes:**
   - **On Each Node Machine:**
     - **Start the Node:** Open a command prompt or terminal, navigate to the Selenium Server jar file's directory, and execute:

       ```
       bash
       CopyEdit
       java -jar selenium-server-standalone-<version>.jar -role node -hub
       http://<hub_ip>:4444/grid/register
       ```

Replace `<hub_ip>` with the IP address of the hub machine.

- **Specify Capabilities (Optional):** You can define specific capabilities for each node, such as the browser type, version, and platform. This ensures that tests are directed to nodes with the appropriate configurations.

5. **Configure the Test Scripts:**
   - **Define Desired Capabilities:** In your test scripts, specify the desired capabilities that match the node configurations.
   - **Initialize RemoteWebDriver:** Use the `RemoteWebDriver` class to connect to the hub:

     ```
     java
     CopyEdit
     DesiredCapabilities capabilities = new DesiredCapabilities();
     capabilities.setBrowserName("chrome");
     capabilities.setPlatform(Platform.WINDOWS);
     WebDriver driver = new RemoteWebDriver(new
     URL("http://<hub_ip>:4444/wd/hub"), capabilities);
     ```

     Replace `<hub_ip>` with the IP address of your hub machine.

6. **Execute Tests:**
   - Run your test scripts. The hub will distribute the tests to the nodes based on the specified capabilities and current availability, allowing for parallel execution.

Benefits of This Configuration:

- **Parallel Test Execution:** Run multiple tests simultaneously across different environments, significantly reducing total execution time.
- **Cross-Browser and Cross-Platform Testing:** Validate application behavior on various browser and OS combinations to ensure compatibility.
- **Scalability:** Easily add or remove nodes to match testing demands without altering the overall infrastructure.

By leveraging Selenium Grid with a hub and multiple nodes on virtual machines, you can create a flexible and efficient testing environment that meets diverse testing requirements.